DATAFLOW-SYNCHRONIZED EMBEDDED FIELD PROGRAMMABLE PROCESSOR ARRAY

The present invention relates to array processors embedded in integrated circuits, such as those implemented in a semiconducting material like silicon, and particularly to reconfigurable embedded array processors.

An embedded system is some combination of hardware or software that is specifically designed for a particular purpose or application within an overall system, and may be fixed in capability or programmable. A mobile phone may, for example, have a power saving integrated circuit (IC) or "chip" operable only with its respective type of phone and devoted exclusively to controlling the display and other elements to conserve power.

The same mobile phone typically includes a digital signal processing integrated circuit, which executes the functions on a digital portion of the radio. In order to adapt to different and/or changing radio broadcast formats of an incoming signal, programmable radios would be desirable. However, digital radio processing functions can entail high data sample rates, along with high computational loads, that are typically impractical to implement on programmable hardware.

Embedded field programmable gate arrays (EFPGAs) are "chip macros" that can be programmable in the field, as well as integrated in a silicon chip, and are available from a limited number of vendors. These special purpose processors operate at high speeds, minimize the amount of hardware required, and minimize software development programming time. Although EFPGAs offer "post silicon" reconfigurability, their design density is poor and their clock speed is unpredictable, particularly for high speed demodulation functions in digital radios.

The present invention is directed to an embedded processor consisting of a two-dimensional array of processing cells and a mechanism for reconfigurably connecting paths between a signal processing circuit and respective cells on a periphery of the array. The processor performs mathematical operations under dataflow control, and is thereby easily integrated within a signal processing circuit operating under the same mode of control. According to this invention the signal processing behavior of the integrated circuit may be reconfigured in the field.

Details of the invention disclosed herein shall be described below, with the aid of the figures listed below, in which same or similar components are denoted by the same reference numbers over the several views:

FIG. 1 depicts an example of a device having an embedded array processor in accordance with the present invention; and

FIG. 2 depicts an exemplary flow of processing in controlling the array processor of FIG. 1; and

FIG. 3 depicts an example of a mixed-signal system on a chip using an embedded array processor according to the present invention.

FIG. 1 shows an exemplary embodiment of an apparatus in accordance with the present invention. A receiver 100, such as one in a broadcast or cable television receiver, local area network wireless receiver or mobile phone receiver, contains an IC 102. The IC 102 includes a system controller 104 and an embedded array processor 106. An array processor is a processor capable of executing instructions that operate on input that may consist of arrays. The embedded array processor 106 has a two-dimensional rectangular array 108 and a mechanism or interface 110 which is shown in FIG. 1 to surround the array 108 on all four edges. The two-dimensional array 108 is composed of processing cells 112.

Preferably, inter-cell connection within the array 108 is such that each cell 112 is connected only to cells 112 whose column is the same and whose row is immediately adjacent, and only to cells 112 whose row is the same and whose column is

immediately adjacent, to realize a "nearest neighbor" connection architecture, as shown in FIG. 2 of commonly owned U.S. Patent Publication No. 2003/0065904, filed October 1, 2001, (hereinafter the '904 application), the entire disclosure of which is incorporated herein by reference. Since inter-cell connection is purely nearest-neighbor, the array offers the flexibility of being scalable.

The interface 110 has border cells 114 connected to each respective processing cell 112 on the periphery of the array 108, each border cell 114 having a buffer 116. The periphery preferably consists of those processing cells 112 which are located on the array edges, i.e., in at least one of the first row, last row, first column and last column. Since internal array connection cell-to-cell, under the nearest neighbor scheme, leaves two neighbors missing for each corner cell 112 and one neighbor missing for each other cell 112 on array edges, the missing connections are each made to a corresponding border cell 114.

Further included in the interface 110 are input/output (I/O) pads 118, one for each border cell 114, and a crossbar network 120 for reconfigurably connecting each I/O pad 118 one-to-one to a corresponding border cell 114. For each such connection an information path is formed. FIG. 1 shows an information path 122 that includes an I/O pad 118 the crossbar network 120 and a border cell 114. Reconfiguring a path causes the path to traverse either a different border cell 114, a different I/O pad 118, or both. The path 124 is a reconfiguration of the path 112 to traverse a different border cell 114.

In a preferred embodiment, the array processor 106 is a systolic processing array, a special-purpose system which can be likened to an assembly line for input operands, although operations typically proceed not in a strictly linear direction but in changing directions. In a two-dimensional array of processing cells, differing mathematical operations are performed on the data by different cells, while data proceeds in an orderly, lock-step progression from one cell to another. An example of a systolic array would be one that multiplies matrices. Entries of a row are multiplied by corresponding entries of a column, and the products are summed to produce an ordered column of sums. Efficiency is achieved by arranging operations to be performed in

parallel, so that the results are produced in the fewest clock cycles. The '904 application provides another example of a systolic processing array, implementing a 32-tap real finite impulse response (FIR) filter. The filter is enhanced by concatenating other levels, two-dimensional and otherwise, to the original two-dimensional array, border cells being connected to processing cells on the periphery of each level. Such an enhanced array, connected by the border cells 114, is also within the intended scope of the present invention.

In one embodiment, the border cells 114 not only provide input to the array 108. They also provide results of array processing to the I/O pads 118. The border cells 114 receive these results by neighbor to neighbor conveyance from the processing cells 112 producing the results. Optionally, the border cell 114 may validate the results and output a data valid signal to the external process.

In a preferred embodiment, the IC 102 includes a memory from which array programs are downloaded by means of a bus to corresponding processing cells 112. The memory is preferably a random access memory (RAM) or other writeable storage device so that updated array programs can be provided, as by an array generator external to the receiver 100.

The system controller 104 passes array programs to a master cell 126 of the embedded array processor 106 over a configuration bus such as the random access configuration bus shown in FIG. 16 of the '904 application. Referring to FIG. 2, the master cell 126 forwards the array programs to the appropriate processing cells 112 (step 202) at system initialization or upon reconfiguration, e.g. implementation of a new algorithm for the processing array 106 (step 204). Due to the parallelism inherent in systolic processing, some of the processing cells 112 may receive identical programs. Alternatively implemented, the system controller 104 and RAM may instead reside within the embedded array processor 106.

Further depicted in FIG. 2 is an exemplary dataflow into the array 108. When a new operand is received on an I/O pad 118, it continues flowing over a path that the crossbar network 120 directs to a corresponding border cell 114 (step 206) which

checks the operand for validity (step 208). If invalid, error processing ensues (step 212), which may involve notifying a user of the receiver 100, and a new operand is requested 216 from the IC application using the embedded array processor 106 (step 216). Alternatively, forward error correction techniques may be applied to rectify the faulty operand. As a further alternative, validation may be performed further upstream, before buffering by the border cell 114. In the embodiment shown in FIG. 2, a valid operand is added to buffer 116 (step 214) and a counter (not shown) is incremented (step 216). Preferably, the buffer cell 116 is implemented to stall the processor providing the new operand when the buffer 116 is full, as by issuing a stall instruction that is routed over the corresponding I/O pad 128 to that processor. A resume instruction is subsequently issued to the processor when an operand is de-buffered. Alternatively, enough buffer space may be provided at the outset to insure that the inflow of new operands in accommodated. In step 218, a parameter corresponding to a predetermined number of input operands is compared to the buffer count. The parameters may vary among border cells 114 and are preferably programmable. The buffers, e.g. ring or circular buffers, are implemented preferably in software. Alternatively, simple first in/first out (FIFO) buffers may be employed.

If the buffer count is greater or equal to the parameter, a trigger is actuated, e.g. the border cell 114 signals the master cell 126 (step 220). If the buffer count is instead less than the parameter control returns to the top of the loop (step 206), and a new operand is awaited.

When an operand is read from the buffer for use by the array 108 (step 222), the counter is decremented (step 224).

The master cell 126, described above regarding its role of distributing downloaded array programs, has the additional role of directing array operations based on the inflow of operands. A new operation to be performed on the array 108, or a new stage of a current operation, may require buffered input operands. When the processing cells 112 needed are idle (step 226), the master cell 126 checks if it has received triggers from all active border cells 114, i.e. the border cells immediately adjacent those of the

needed processing cells on the array periphery (step 228). If all of the triggers have been received, or when this occurs, the operands are read from buffer, the new operation or stage is commenced and the triggers are reset (step 230).

In accordance with the above-described border and master cell protocol, the array processor 106 performs mathematical operations whose timing is based on a flow of input operands along the paths providing the operands to the array 108.

In a preferred embodiment, the parameter for step 218 is set to zero. In effect, a Kahn process network is therefore implemented. In such a network the processors are interconnected by channels having first-in/first-out (FIFO) buffers. A processor can either send data to a FIFO channel, or else receive data from a FIFO channel. If a processor requests a read and no data is available then the processor stalls until the data is available. In a pure Kahn process network enough buffer space is provided to accommodate an unlimited number of write operations. In the current implementation, writes are preferably limited so that if a processor writes to a FIFO channel and it is full then the processor stalls until there is room to write.

As one example of the current invention, other processors on the IC 102 may, along with the embedded array processor 106, form a Kahn process network with bounded writes, i.e. writes that are stalled when the buffer is full. The buffers 114 are each implemented as a pair of FIFOs.

In this preferred embodiment, step 216 can be retained to detect when the buffer 114 is full, at which point a stall instruction as described above is preferably issued to the processor providing the input operands. If step 216 is retained, the counter decrementing process (steps 222, 224) for the border cells would be retained as well, and a resume instruction would issue when an operand is de-buffered.

Array programs may be prepared using a graphical user interface (GUI) that can edit and show the code to be downloaded to RAM on the IC 102 and then to each programming cell 112.

The embedded array processor 106 is particularly useful for integration, in a manner similar to that of embedding an FPGA within a system on chip (SoC). The

border cell-based interface 110 affords simple integration and a simple software programming flow in place of the proprietary hardware design flow characteristics of EFPGAs.

As illustratively depicted in FIG. 3, the embedded array processor 106 may be integrated with a general system on a chip 102 that includes a digital circuit 302 and possibly an analog circuit 304, in order to introduce reconfigurability within the system. The digital circuit may be composed of fixed design, digital circuit modules 306. One of the modules 306 may act as the system controller 104. The modules 306 have pins interconnected by routing switches 308, which normally connect the outputs of one digital circuit module 306 to the input of another. The routing switches 308 are also capable of replacing the connection between two modules 306 with an alternative input and output connector pair 310 to switch connection from one or both of the two modules 306 to a respective pin 128 of the embedded array processor 106. The digital circuit may also be integrated with the analog circuit 304 using one or more analog-to-digital converters 314 to convert the analog signals from the analog circuit outputs 304 to digital signals to be connected routed to the digital circuit modules 306. In a similar way digital circuit outputs to the analog circuit 304 may be converted from digital samples to analog signals by a digital-to-analog converter 316. A routing switch 318 may also be placed between the converter 314 and the digital circuit 302 in order to afford switchable connection from and to the processor 106. In particular, the input/output connector pair 320 affords switching between a signal pathway from the analog circuit to the digital circuit and a signal pathway to or from said one or more input/output pads. Similarly, a routing switch 322 may be placed between the digital-to-analog converter 316 and the digital circuit 302. The routing switches 308, 318, 322 in combination with the reconfigurable interface 110 of the processor 106 provide the analog and digital circuits 302, 304 with one or more dataflow-driven signal processing functions into the array processor 307 and insert such functions into either the chain of the digital circuit. In a similar fashion it is possible to program a dataflow-driven signal processing function into the array processor 307 and insert such functions into the analog circuit 301. As seen in

FIG. 3, the processor array 106 may interface with a plurality of inhomogeneous parallel processing elements on a chip. The intended scope of the invention is not limited to the configuration shown and may include, for example, alternative and/or additional connections among the integrated circuit elements.

While there have been shown and described what are considered to be preferred embodiments of the invention, it will, of course, be understood that various modifications and changes in form or detail could readily be made without departing from the spirit of the invention. For example, reconfigurable routing can be accomplished via a local selection mechanism in each border cell, rather than by a crossbar network. It is therefore intended that the invention be not limited to the exact forms described and illustrated, but should be constructed to cover all modifications that may fall within the scope of the appended claims.